Sparse quantum Gaussian processes to counter the curse of dimensionality

Kuś, Gaweł I.; van der Zwaag, Sybrand; Bessa, Miguel A.

DOI
10.1007/s42484-020-00032-8

Publication date
2021

Document Version
Final published version

Published in
Quantum Machine Intelligence

Quantum Machine Intelligence (2021) 3: 6
https://doi.org/10.1007/s42484-020-00032-8

RESEARCH ARTICLE

Sparse quantum Gaussian processes to counter the curse of dimensionality

Gaweł I. Kuś[1] · Sybrand van der Zwaag[1] · Miguel A. Bessa[2]

Received: 22 June 2020 / Accepted: 2 December 2020 / Published online: 17 February 2021

Abstract

Gaussian processes are well-established Bayesian machine learning algorithms with significant merits, despite a strong limitation: lack of scalability. Clever solutions address this issue by inducing sparsity through low-rank approximations, often based on the Nystrom method. Here, we propose a different method to achieve better scalability and higher accuracy using quantum computing, outperforming classical Bayesian neural networks for large datasets significantly. Unlike other approaches to quantum machine learning, the computationally expensive linear algebra operations are not just replaced with their quantum counterparts. Instead, we start from a recent study that proposed a quantum circuit for implementing quantum Gaussian processes and then we use quantum phase estimation to induce a low-rank approximation analogous to that in classical sparse Gaussian processes. We provide evidence through numerical tests, mathematical error bound estimation, and complexity analysis that the method can address the "curse of dimensionality," where each additional input parameter no longer leads to an exponential growth of the computational cost. This is also demonstrated by applying the algorithm in a practical setting and using it in the data-driven design of a recently proposed metamaterial. The algorithm, however, requires significant quantum computing hardware improvements before quantum advantage can be achieved.

Keywords Gaussian processes · Low-rank approximation · Design of materials · Data-driven design

1 Introduction

Gaussian processes (GPs) are a Bayesian machine learning method capable of inference from noisy data while quantifying uncertainty (Williams and Rasmussen 2006; Seeger 2004; Murphy 2012). They are used in classification and regression tasks (Williams and Rasmussen 2006; Seeger 2004; Murphy 2012), and also in Bayesian optimization (Shahriari et al. 2015; Acerbi and Ji 2017) where a trade-off between exploration and exploitation is sought based on their uncertainty quantification capabilities.

Among many domains of application (Belgacem et al. 2020; Li et al. 2020; Czekala et al. 2017; Ažman and Kocijan 2007; Bessa et al. 2019), the advantages of GPs over other machine learning methods are well illustrated by the data-driven design and optimization of materials (Bessa et al. 2019; Tancret et al. 1999; Lookman et al. 2017; Frazier and Wang 2016; Bessa et al. 2017) and structures (Bessa and Pellegrino 2018) with uncertain behavior. In these problems, GPs obtain robust and realistic solutions by estimating uncertainty, avoiding overfitting and dealing with imperfect data. However, data-driven solutions using GPs are limited to relatively small datasets (a few thousands of data points) due to the computational cost involved in Bayesian inference. Unfortunately, this limitation cripples the application of GPs to problems involving a large number of parameters due to the "curse of dimensionality," i.e., each new design parameter implies an exponential growth of the design space (exponentially larger datasets).

More scalable Bayesian machine learning methods exist, such as Bayesian neural networks and the more scalable alternative of ensembles of neural networks (Wilson and Izmailov 2020; Pearce et al. 2018), but they also have limitations not present in GPs. GP models are easier to train and interpret (Williams and Rasmussen 2006; Seeger 2004; Tancret et al. 1999) because they have fewer hyperparameters and the optimization benefits from convexity of the GP posterior (Williams and Rasmussen

✉ Miguel A. Bessa
M.A.Bessa@tudelft.nl

[1] Novel Aerospace Materials, Faculty of Aerospace Engineering, Delft University of Technology, Delft, The Netherlands

[2] Materials Science and Engineering, Delft University of Technology, Delft, Netherlands

2006). Conversely, training neural network is an NP-complete problem for both proper (Blum and Rivest 1992; Bartlett and Ben-David 2002) and improper (Daniely et al. 2014) learning approaches. In practice, training neural network is hampered by ubiquitous local minima because the typical procedure involves random initialization of a very large number of parameters that are fitted by minimizing a generally non-convex loss function using gradient descent approaches (Geiger et al. 2020; Lee et al. 2019).

In this work, we address the problem of scalability of GPs, therefore enlarging their applicability and making them competitive with traditionally more scalable methods like neural networks. In particular, we consider a recently proposed quantum algorithm for GP regression (Zhao et al. 2019a) and explore enhancing its computational performance by incorporating classical concepts for improving GP scalability, specifically through inducing sparsity in GP model via low-rank approximation.

Formally, given a training dataset $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\}$ consisting of n d-dimensional input points $\mathbf{X} = \{\mathbf{x}_i \in R^d\}_{i=1}^n$ and n corresponding outputs $\mathbf{y} = \{y_i = y(\mathbf{x}_i) \in R\}_{i=1}^n$, GP regression provides a non-parametric approach to inference of a latent function $f(\mathbf{x})$ based on the observe value y affected by noise such that $y = f(\mathbf{x}) + \epsilon$ and $\epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2)$, where σ_ϵ^2 is the noise variance. Note that this article follows a notation where matrices are referred to by bold capital letters, vectors by bold small case letters, and scalars by letters that are not in bold. The inference relies on the definition of GPs, which specifies them as a collection of random variables following a joint multivariate normal distribution (in this case the random variable is the value of the function at a given point). This definition gives rise to a prior distribution where both observed and unobserved values of the inferred function (y and f_* respectively) follow a joint multivariate distribution, completely specified by mean and variance:

$$\begin{bmatrix} \mathbf{y} \\ f_* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} \mathbf{K} + \sigma_\epsilon^2 \mathbf{I} & \mathbf{k}_* \\ \mathbf{k}_*^T & k_{**} \end{bmatrix}\right) \tag{1}$$

In this case, the mean is conventionally assumed to be 0 for convenience. $\mathbf{K}$ is a $n \times n$ covariance matrix obtained by calculating covariance between the training points $\mathbf{X}$, e.g., $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ is the element (i, j) of the matrix $\mathbf{K}$, and $\mathbf{I}$ is a $n \times n$ identity matrix. Similarly, $\mathbf{k}_*^T$ is a $1 \times n$ vector obtained by calculating covariance between the training points $\mathbf{X}$ and each test input point $\mathbf{x}_*$, i.e., $\mathbf{k}_* = \{k(\mathbf{x}_1, \mathbf{x}_*), \ ..., \ k(\mathbf{x}_n, \mathbf{x}_*)\}$, and k_{**} by calculating covariance between each test input, i.e., $k_{**} = k(\mathbf{x}_*, \mathbf{x}_*)$. The kernel function usually defines similarity between the points and controls smoothness of GP. For instance, the most common kernel function is the Radial Basis Function (RBF):

$$k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 exp\left(\frac{|\mathbf{x} - \mathbf{x}'|^2}{2l^2}\right) \tag{2}$$

Covariance functions often also depend on additional free parameters, e.g., for the RBF function the signal variance σ_f^2 and the length-scale l. Those hyperparameters are usually tuned by maximizing the log of marginal likelihood (Williams and Rasmussen 2006).

Following the approach of Bayesian inference, the prior distribution is then conditioned on the observed data to yield the posterior distribution specified by mean $\mu(\mathbf{x}_*)$ and variance $\sigma^2(\mathbf{x}_*)$ as:

$$\begin{aligned} \mu(\mathbf{x}_*) &= \mathbf{k}_*^T \left(\mathbf{K} + \sigma_\epsilon^2 \mathbf{I}\right)^{-1} \mathbf{y} \\ \sigma^2(\mathbf{x}_*) &= k_{**} - \mathbf{k}_*^T \left(\mathbf{K} + \sigma_\epsilon^2 \mathbf{I}\right)^{-1} \mathbf{k}_* \end{aligned} \tag{3}$$

The mean can be used to predict values of the latent function at some new test point $\mathbf{x}_*$, while variance $\sigma^2(\mathbf{x}_*)$ provides with an estimate of the corresponding uncertainty. As can be seen from this expression, calculating the posterior mean and variance requires inversion of the full covariance matrix $\left(\mathbf{K} + \sigma_\epsilon^2 \mathbf{I}\right)$ of size $n \times n$, where n is the number of training points. This is the origin of the scalability issues of GPs, as in general this operation scales as $O(n^3)$.

Recent efforts have focused on improving the scalability of GPs by inducing sparsity in GP models (Quiñonero-Candela and Rasmussen 2005; Snelson and Ghahramani 2006; Titsias 2009; Chalupka et al. 2013; Hensman et al. 2013; Hensman et al. 2015; Liu et al. 2018). In this case the term "sparsity" is used by machine learning community to refer to approximate GP models, where only a subset of latent variables is given exact treatment (Quiñonero-Candela and Rasmussen 2005), with the aim to reduce the cost of inverting the kernel matrix. The simplest approach to realizing GP sparsity is by selecting only a subset of data points $m \ll n$. This reduces the size of the kernel matrix to $m \times m$ and the complexity to $O(m^3)$ (Quiñonero-Candela and Rasmussen 2005; Chalupka et al. 2013). However, finding a suitable subset is a non-trivial problem (Quiñonero-Candela and Rasmussen 2005).

Alternatively, effective sparse GP models are obtained by relying on low-rank representations of the kernel matrix (Quiñonero-Candela and Rasmussen 2005; Liu et al. 2018). The Nystrom approximation is particularly suitable as it provides an approximation (rank m) in a form: $\mathbf{Q}_{n\times n} = \mathbf{K}_{n\times m}\mathbf{K}_{m\times m}^{-1}\mathbf{K}_{n\times m}^T$ which allows for inversion with the Sherman-Morrison-Woodbury formula, reducing complexity from $O(n^3)$ to $O(nm^2)$ (Williams and Seeger 2001; Kumar et al. 2009). Consequently, satisfying $m \ll n$ allows for significant speed-ups (Williams and Seeger 2001). Yet, the key to achieving high-quality approximations is in the construction of the inner covariance matrix $\mathbf{K}_{m\times m}$. Different methods were proposed

(Williams and Seeger 2001; Wilson and Nickisch 2015), but one of the most relevant relies on introducing a set of $m \ll n$ artificial points, called inducing points (Titsias 2009; Snelson and Ghahramani 2006; Quiñonero-Candela and Rasmussen 2005). The position of the inducing points is optimized akin to other GP hyperparameters, which allows for achieving high-quality approximations. This concept gives rise to two significant sparse GP methods: variational free energy (Titsias 2009) and fully independent training conditional (Snelson and Ghahramani 2006).

Sparse Gaussian processes, however, provide only limited solutions for countering the curse of dimensionality. The number of inducing points, which determines the speed-up, strongly affects the quality of the approximation. Consequently, if the number of inducing points is not sufficient, the quality deteriorates quickly. Therefore, the speed-up does not match the exponential growth of the design space. A different pathway was revealed recently by Zhao et al. (2019a) where the scalability of Gaussian processes is addressed with quantum computing.

In this article, we discuss the quantum algorithm for Gaussian processes (QGP) introduced by Zhao et al. (2019a) and propose a modification of the quantum phase estimation routine to induce a low-rank approximation, which gives rise to a sparse QGP model (SQGP). We demonstrate our implementation of a sparse variant of the QGP algorithm (SQGP), analyze its accuracy and scalability, and compare to the classical alternatives: Gaussian processes, sparse Gaussian processes, and Bayesian neural networks (in particular, the scalable ensembling method by Pearce et al. (2018)). The primary contribution of this work is to present a mechanism for inducing a low-rank approximation (SQGP) in the original QGP algorithm by introducing a cutoff in the eigenspectrum. This is achieved by exploiting finite resolution of QPE such that the smallest eigenvalues which fall below a carefully selected resolution threshold are approximated as 0. We note that this effectively realizes a quantum equivalent of principal component analysis, which is known to be an optimal low-rank approximation. We address the relation of SQGP to the classical low-rank approximations used for sparse GP methods, in particular the Nystrom method which stands behind the most effective classical sparse GP methods. With numerical experiments and error bounds, we demonstrate that QGP and its sparse version (SQGP) are not only more scalable and accurate than classical sparse Gaussian processes, but they can also outperform classical Bayesian neural networks.

We note that despite current quantum computing hardware not being sufficiently developed to experimentally show quantum advantage for large datasets (Zhao et al. 2019b), this article ends by demonstrating the feasibility of SQGP in the design of a recently presented supercompressible metamaterial (Bessa et al. 2019). The results show great promise.

2 Inducing sparsity in quantum Gaussian processes

The QGP algorithm proposed by Zhao et al. (2019a) relies on a common concept of quantum machine learning, where quantum computing is applied to speed-up linear algebra operations in classical machine learning (Biamonte et al. 2017). In particular, QGP employs the well-known HHL algorithm (Harrow et al. 2009) to accelerate the matrix inversion in GP regression. The concept was later extended also to a continuous variable setting by Das et al. (2018). The algorithm is formulated to solve a problem posed in the form $q = \mathbf{u}^{\mathbf{T}}\mathbf{A}^{-1}\mathbf{v}$. As shown with the schematic circuit in Fig. 1, the procedure is carried out in three steps: (1) initialization of the input **vectors** u and v in superposition, (2) matrix inversion with controlled HHL, and (3) applying the measurement operator to efficiently retrieve the result in the form of $\mathbf{u}^{\mathbf{T}}\mathbf{A}^{-1}\mathbf{v}$. This allows to solve (3), where the posterior mean can be found by substituting $\mathbf{u} = \mathbf{k}_*$, $\mathbf{A} = (\mathbf{K} + \sigma_n^2\mathbf{I})$ and $\mathbf{v} = \mathbf{y}$, and similarly for the posterior variance.

In this paper, we implement and analyze Zhao's QGP algorithm (Zhao et al. 2019a) to reveal a natural yet unexplored way of inducing a low-rank approximation (analogous to that in classical sparse GPs) by controlling the quantum phase estimation (QPE) subroutine used by the HHL algorithm for system diagonalization (Harrow et al. 2009). QPE is an approximate algorithm (Nielsen and Chuang 2000) that encodes the eigenvalues as a finite binary expansion with k digits precision (where k is the number of qubits in the eigenvalue register). This effectively discretizes the eigenspectrum with a finite resolution determined by k. Since k qubits allow to resolve 2^k values in a range $[0, \lambda_{\max}]$, the minimum non-zero eigenvalue that can be resolved is $\lambda_{\min} = \frac{\lambda_{\max}}{2^k-1}$ (see also the "Eigenspectrum cut-off strategy" section in the Supplementary Information). All eigenvalues below this threshold are approximated as 0, which results in a cut-off of the corresponding eigenmodes. This cut-off associated to the number of qubits k creates a low-rank approximation of the input matrix, $\mathbf{A}_{n\times n} \approx \mathbf{U}_{n\times m}\mathbf{\Lambda}_{m\times m}\mathbf{U}^T_{n\times m}$, where m corresponds to the number of resolved eigenmodes. This is equivalent to principal component analysis (PCA) with the only difference that the eigenvalues are resolved with finite accuracy determined by k. Therefore, QPE with limited k can effectively realize quantum principal component analysis (qPCA) (yet in a different setting that the commonly known qPCA algorithm proposed by Lloyd et al. (2014)).

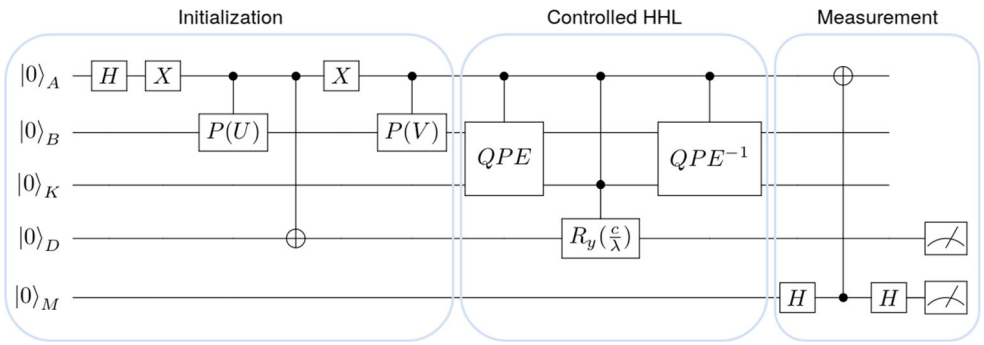

Fig. 1 Schematic circuit of the QGP algorithm

This feature can be used for fixing rank of ill-conditioned systems to perform matrix inversion on a subspace (Harrow et al. 2009; Wossnig et al. 2018). However, the above observation has also important implications for the QGP algorithm because QPE within the HHL subroutine can be the source of a low-rank approximation if the number of qubits k is appropriately chosen, to yield SQGP model. Note that the term "sparsity" is used in the machine learning community with a broader sense than the exclusive exploration of sparse matrices. Instead, the sparsity in GP models arises from an approximate treatment of latent variables, which allows to address the scalability issue created by inverting the covariance matrix. As mentioned in Section 1, in practice, this is achieved by relying on low-rank approximations. In other words, QPE (or equivalently quantum PCA) can have a similar effect as the above-mentioned classical Nystrom approximation which induces sparsity in classical Gaussian process models to improve scalability of the method. Yet, this raises an important question: why is PCA not already used as a means to induce sparsity in Gaussian processes with classical computers?

Constructing a low-rank approximation with PCA for GPs with classical computing is not beneficial because PCA requires eigendecomposition of the system and in classical computing this operation scales similarly to matrix inversion ($O(n^3)$). So, performing classical PCA to approximating the covariance matrix to obtain sparse GPs is as computationally expensive as inverting the complete matrix to obtain full GPs. Interestingly, this is not necessarily the case in quantum computing—given access to data loaded in a state vector (e.g., with qRAM (Giovannetti et al. 2008)) the eigendecomposition can be carried out exponentially faster than classical counterparts (Biamonte et al. 2017). Achieving exponential speed-ups, however, is not sufficient if the accuracy of the method deteriorates with an increase in the number of training points. Therefore, the following sections focus on both aspects to assess the merits and pitfalls of QGP and SQGP when compared with the classical Bayesian machine learning algorithms (classical GPs and Bayesian neural network ensembles).

3 Comparative scalability analysis

This section expands on work of Zhao et al. (2019a) with an analysis of the effects of inducing low-rank approximation on the scalability of QGP. For this purpose, we devise a cost model that quantifies circuit depth with respect to three parameters: the size of eigenvalue register k, the number of time slices r controlling the approximation of the Hamiltonian simulation in QPE, and system size n (2, 4, 8 and 16). Based on the complexity of each routine in the implemented quantum circuit, the cost function with respect to three parameters (k, r, n) is assumed as follows:

$$d = \alpha_1 kr \log(n) n^2 + \alpha_2 \cdot k + \alpha_3 \cdot k^2 + \alpha_4 \cdot 2^k \tag{4}$$

where α_i are scaling factors to be determined for each term. The first term corresponds to Hamiltonian simulation which exponentiates input matrix H in order to provide unitary operator e^{iHt} for QPE routine. In our implementation, the exponentiation is carried out with Suzuki-Trotter scheme (Hatano and Suzuki 2005) given by:

$$e^{(B+C)t} = \left(e^{B\frac{t}{r}} e^{C\frac{t}{r}}\right)^r + O\left(\frac{1}{r^2}\right) \tag{5}$$

where B and C are some arbitrary operators with some commutation relation $[B, C] \neq 0$, and $\left(\frac{1}{r^2}\right)$ is an error arising due to approximating the exponentiation of non-commuting terms with a product formula. This allows to approximate an evolution of a Hamiltonian $H = \sum_i H_i$ by evolving local Hamiltonians H_i over a number of r smaller time steps $\left(\frac{t}{r}\right)$. The second and the third terms of the model (4) correspond to Fourier transform which follows after matrix exponentiation in QPE, and scales quadratically with number k of ancilla qubits in the eigenvalue register. The fourth term accounts for finding reciprocals of eigenvalues, necessary to perform controlled rotation in HHL: $R_Y(\frac{1}{\lambda})$ on the ancilla qubit. Numerical experiments discussed in the Supplementary Information ("QGP cost model" section) demonstrate the adequacy of Eq. 4 in predicting the circuit depth. Note that we are assuming a conservative estimate for the Hamiltonian simulation term, as we are

considering a scaling of $O\left(n^2 \log(n)\right)$ due to limitations of the implementation, but in theory it can become $O\left(\log(n)\right)$. In addition, the state preparation step was neglected as suggested in other works because this step is assumed to be carried out efficiently using qRAM (Harrow et al. 2009; Zhao et al. 2019a) or alternative approaches (Zhao et al. 2018). The measurement operator is disregarded because it is implemented with only three gates, which is negligible compared to other terms.

Observing (4), we note that k has an important contribution to the circuit depth, which is directly related to the computational cost of the algorithm. The number of qubits in eigenvalue register, the parameter k, controls the resolution of eigenvalues in QPE and determines the cut-off threshold in qPCA approximation. The equation shows that the overall algorithm depth is dominated by the Hamiltonian simulation (first term) for large systems, but it also shows that despite the other terms being less relevant the influence of k on these terms is also stronger. Hence, inducing sparsity by reducing k can have a significant impact on computational cost. Note that the overhead terms, i.e., excluding the Hamiltonian simulation term, can have a significant contribution to the overall cost of the algorithm. For instance, matching the accuracy typical in classical computing, e.g., $O(10^{-15})$ for Numpy (van der Walt et al. 2011), would require $k = 50$ that would lead to $O(10^{16})$ gates of overhead (i.e., depth on top of the Hamiltonian simulation cost). Therefore, inducing sparsity provides a way for drastic reduction of this overhead by several orders of magnitude, which might be a critical factor for applicability of the algorithm on limited hardware.

Figure 2 illustrates the scalability of QGP and SQGP compared with state-of-the-art classical alternatives. As mentioned previously, sparse GPs scale as $O(nm^2)$, which for $m \ll n$ provide significantly better scalability as opposed to $O(n^3)$ of the full GP models. However, the choice of suitable m (which provides sufficient performance) is dependent on both the problem and the particular sparse GP method. Therefore, to account for those variations, in Fig. 2, we illustrate scalability of sparse GPs as a range of values. In the least favorable scenario (for non-trivial functions), the number of inducing points scales linearly with n, i.e., $m = \beta n$. In this case, inducing sparsity of the GP model allows for only a constant offset over the full GP, yielding $O(\beta^2 n^3)$, where $\beta < 1$. This is illustrated in Fig. 2 with an upper limit, which was plotted assuming $\beta^2 = 0.05$ (note that this is not a rigorous bound). On the other hand, some functions are relatively simple, in which case the dataset can be well approximated with only a few inducing points. We assume that in the best-case scenario m scales logarithmically with n (for illustration assumed $m = log_{10}(n)$), which is plotted as the lower limit in Fig. 2.

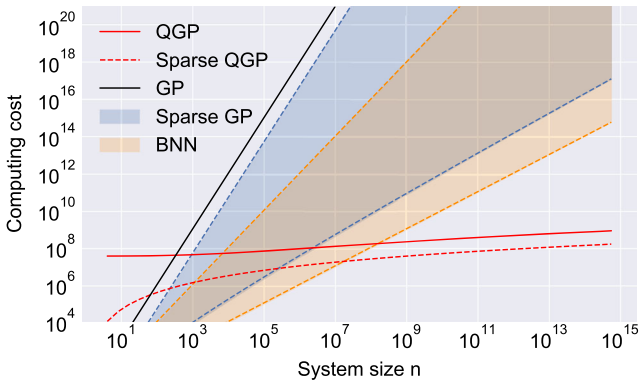

Fig. 2 An illustrative comparison of scalability of different machine learning methods. A comparison of classical full and approximate Gaussian processes (GP), full and sparse quantum Gaussian processes (QGP), and Bayesian neural network (BNN) ensembles

We also include the scalability of Bayesian neural network (BNN)—a classical machine learning model capable of uncertainty quantification that is an alternative to GPs. In particular, we consider the method proposed by Pearce et al. (2018), which approximates Bayesian posterior using ensembles of neural networks, providing the most scalable alternative (its scaling is comparable to that of standard neural networks, with only a multiplicative factor overhead due to ensembling). Notwithstanding that training neural networks is an NP-complete problem in theory, in practice, training in polynomial time provides satisfactory results (Blum and Rivest 1992; Livni et al. 2014). In addition, the computational cost of training neural networks is greatly dependent on their architecture; for instance, using the popular backpropagation algorithm for a dataset of n points with m features and an architecture of k hidden layers, each with h neurons with o output neurons and training over i iterations, leads to a scaling of $O(n \cdot m \cdot h^k \cdot o \cdot i)$ (Pedregosa et al. 2011). Also note that deep architectures used for large datasets are usually trained in an over-parametrized regime, i.e., the number of parameters N is much larger than the number of datapoints n ($N >> n$) (Geiger et al. 2019). Recently, it was found that the best generalization performance can be reached for N just beyond a value N^* for which so-called jamming transition occurs (Geiger et al. 2019; Geiger et al. 2020). Based on this, it was found that in general for random data $N \approx O(n)$ (Geiger et al. 2019); therefore, it can be assumed that the simplest neural network with a single hidden layer requires $h \approx O(n)$, and the overall training will scale as $O(n^2)$. However, for structured data, it can be expected that N might be sub-linear with n (Geiger et al. 2019) and the neural networks could reach overall $O(n)$ scaling. To illustrate the possible performance variations, similarly as in sparse GPs, the cost of BNNs is shown in Fig. 2 as varying between the two estimated limits.

The difference between full-rank and low-rank QGP is significant especially at lower system sizes n, as shown in Fig. 2, as the overhead terms dependent on k are larger in magnitude than those dependent on system size n, and thus dominate the overall cost. For larger system sizes n, however, the cost term dependent on n takes over. Consequently, in this regime, the cost of low-rank QGP is reduced roughly proportionally to the difference in k between QGP and SQGP (see the cost model (4)) which on the log-scale plot in Fig. 2 can be seen as an approximately constant offset. Nevertheless, despite a relatively modest cost saving (up to a few orders of magnitude), due to the logarithmic scaling (revealed as an almost flat slope of the curve in Fig. 2), inducing sparsity allows for SQGP to tackle dramatically larger systems compared to QGP, given the same cost. Note that the offset between QGP and SQGP can be increased further by lowering k, thus leading to further gains. This indicates practical advantage of SQGP over the full QGP, especially on a limited hardware.

Our analysis provides a qualitative comparison of scaling based on estimated complexity rather than actual cost. It demonstrates that QGP and SQGP can outperform all classical methods considered herein, including Bayesian neural networks ensembles. Yet, the usefulness of a new method also hinges on its accuracy, not just its efficiency.

3.1 Accuracy of QGP and SQGP

Since sparsity is induced in QGP by quantum PCA, and knowing that classical PCA is the best low-rank approximation (Murphy 2012), we expect that SQGP provides a better approximation than classical sparse GPs which rely on other low-rank approximation methods. We verify that by estimating error bounds for the two different methods. The derivation of the various bounds is found in the respective section of the Supplementary Information.

In classical PCA, the error of approximating matrix $\mathbf{A}$ with a matrix $\widetilde{\mathbf{A}}_m$ of rank m can be roughly estimated as:

$$\epsilon = \left\| \mathbf{A} - \widetilde{\mathbf{A}}_m \right\|_F \approx \lambda_{m+1} \tag{6}$$

where λ_{m+1} is the first truncated eigenvalue. Since the accuracy in quantum PCA differs from the classical one by the finite resolution of eigenvalues, the best case scenario is that all the resolved eigenvalues in qPCA can be resolved exactly (i.e., no error due to finite precision) and the overall error of the approximation is the same error as the error of the classical PCA, as in Eq. 6. In this case, the eigenvalue λ_{m+1} is bounded by the resolution limit $\lambda_{m+1} \leqslant \frac{\lambda_{\max}}{2^k-1}$, which yields the following lower bound:

$$\left\| \mathbf{A} - \widetilde{\mathbf{A}} \right\|_F \geqslant \frac{\lambda_{\max}}{2^k - 1} \tag{7}$$

On the other hand, the worst-case scenario contemplates that every eigenvalue that contributes to the quantum PCA approximation is resolved with maximum error. This, combined with the error of truncating the eigenspectrum Eq. 7, yields the following upper bound:

$$\left\| \mathbf{A} - \widetilde{\mathbf{A}} \right\|_F \leqslant \lambda_{\max} \left(\frac{1}{2^k - 1} + \frac{\sqrt{m}}{2^{k - log(2+1/(2\delta))} - 1} \right) \tag{8}$$

The bound holds with probability $1 - \delta$. For comparison, the bounds obtained for classical sparse GPs using the Nystrom approximation can be derived based on the error analysis of Wang and Zhang (2014). As mentioned, the details are provided in the Supplementary Information.

When comparing the error bounds of the Nystrom approximation and QPE, we note that there is an overlap. The lower bound for the Nystrom approximation is above the lower bound for QPE, which indicates that QPE can outperform the classical method. However, the lower Nystrom bound is below the upper bound of QPE, so the bounds do not allow drawing general conclusions about the comparative performance of the two methods. In addition, the bounds obtained for SQGP and classical sparse GPs cannot be compared directly due to the dependence on different parameters that are not linearly related to each other: in QPE, the eigenmodes are discriminated by the threshold value controlled by k which is not necessarily the same as m, while in the Nystrom approximation the number of eigenmodes is directly m. We illustrate the truncation of the eigenspectrum by the two methods in section "Eigenspectrum cut-off strategy" of the Supplementary Information, and in Fig. 3 we show the bounds and a typical result from a numerical experiment where a number of sample covariance matrices were approximated using the two methods. The tests were carried out using several sets of testing matrices and other cases can be seen in the Supplementary Information as well.

In general, the results for most of the tested systems were found to be very similar to the ones shown in Fig. 3. The only exceptions were a few systems with artificially mild eigenspectrum decay. Those systems, however, are not expected to provide a good representation of real examples. For most of the systems, the eigenspectrum is characterized by fast decay of eigenvalues. The literature suggests (Wathen and Zhu 2015) an exponential decay of the eigenspectrum, but we found a decay closer to a double exponential: $\lambda_i = \lambda_{\max} R^{i^b}$, where $R < 1$ and $b \approx 2$. This combined with floating point precision results in eigenspectrum characterized by two domains as shown in the figure. One region with relatively few large eigenvalues follow the double exponential decay, and a plateau of small eigenvalues (caused by finite precision of number representation). Direct comparison of the two methods required finding the $k - m$ relationship.

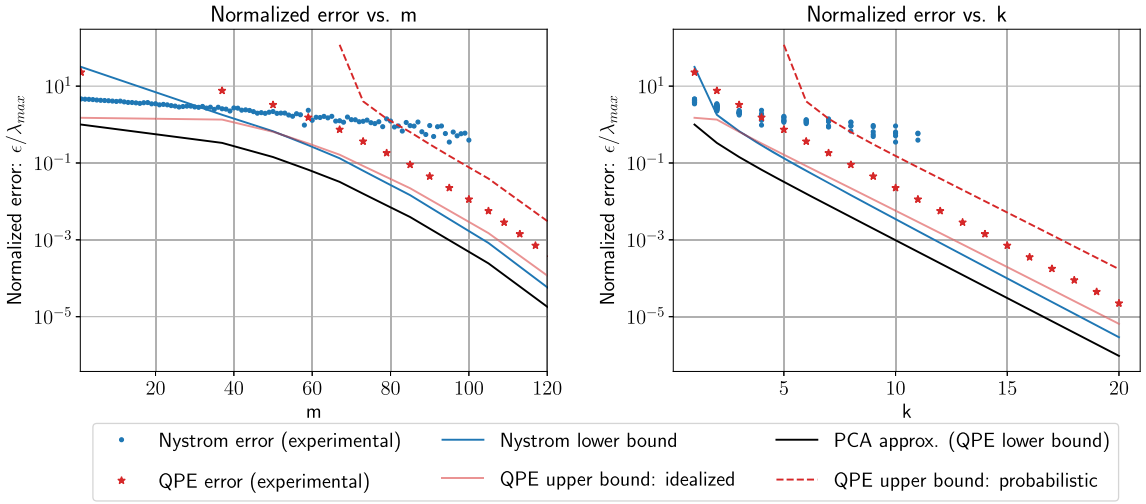

Fig. 3 Numerical verification of error bounds of classical and quantum GPs. The bounds are derived for the two different low-rank approximation methods considered herein: classical Nystrom scheme and quantum PCA

This is done numerically by counting the number of resolved eigenmodes (m) included in the quantum PCA approximation for each tested number k, as shown in the figure in lower right corner.

The theoretical bounds are verified by comparing with the actual errors obtained with the numerical experiments. In general, the errors of QPE follow well within the bounds. This is also the case for Nystrom method. However, here, the bounds are considerably more loose. In particular, the upper bound is very conservative (the value is too large to be seen in the figure), which is an effect of inequalities used in the derivation.

Comparing the numerical errors obtained with the two approximation methods, shown in upper figures, it turns out that qPCA provides a better approximation, regardless whether the comparison is done with respect to parameter m or k. The accuracy of Nystrom method is considerably worse, especially for cases where QPE reaches higher resolution (large k). The only exception is for cases with very small eigenvalue resolution ($k < 5$).

3.2 Cost-accuracy trade-off

Finally, we compare numerically the cost-accuracy trade-off of the classical sparse Gaussian processes against the sparsity induced via qPCA. For this purpose, a small numerical example was set up, where the two methods were employed to approximate a full GP model inferring a function from a small noisy dataset of 50 points. The results are presented in Fig. 4. The accuracy of the approximations was quantified by mean squared error (MSE) of the discrepancy between the approximate and the full GP model. In sparse GP, the accuracy was improved by increasing the number of inducing points, while in qPCA the eigenvalue resolution was increased (via parameter k). Those two parameters were then related to the increase in computational cost via complexity terms.

Classical sparse GP reaches a much better accuracy compared to what could be expected from the testing of the Nystrom approximation. This, however, is to be expected, as in sparse GP the approximation is improved by optimizing the locations of the inducing points. Nevertheless, the cost of improving accuracy of sparse GP Nystrom approximation scales quadratically with the number of inducing inputs m, while qPCA allows for similar accuracy with only a linear increase in cost. We note that the dip in Fig. 4 for classical sparse GPs between values of 100 and 1000 of the relative cost increase factor is an effect of random fluctuations in MSE due to the stochastic nature of the optimizer.

GPs were also compared with neural networks, which classically provide a more scalable alternative (for details

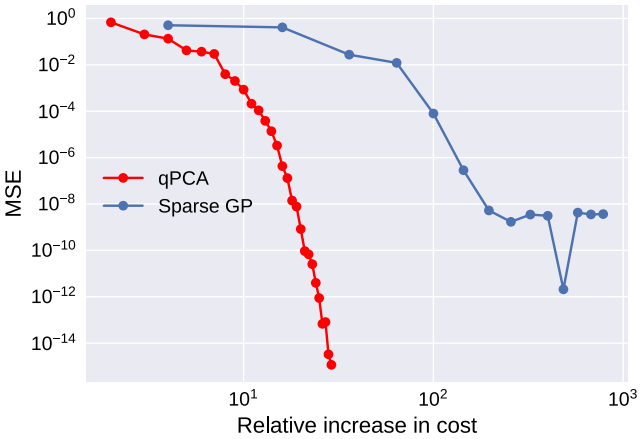

Fig. 4 Cost-accuracy trade-off for classical and quantum sparse GPs. The classical sparse Gaussian processes with optimized locations of inducing points are compared numerically against the SQGP with sparsity induced via qPCA

see the "Comparison with neural networks" section of Supplementary Information). However, as the performance of neural networks is strongly dependent on the architecture and training parameters, several different configurations were considered, but we achieved limited accuracy when comparing with full Gaussian process model due to overfitting. The problem could be addressed by adding regularization, but this highlights a key issue with neural networks: training is not trivial, and the performance is strongly dependent on user's experience.

4 SQGP in data-driven design of materials

In the first part of this paper, we discussed inducing sparsity in the quantum algorithm for Gaussian processes and showed that it can outperform any comparable classical algorithms including Bayesian neural networks and sparse Gaussian processes. In this part, we argue that scalability of sparse QGP allows overcoming the dimensionality problem in data-driven design of materials. To support our hypothesis, we demonstrate a proof-of-concept example of an application in data-driven optimization of material design.

The example problem is based on a recent design of a metamaterial unit cell (Bessa et al. 2019), shown in Fig. 5. The architecture of this metamaterial exploits coiling of the unit cell to achieve supercompressibility even when fabricated with intrinsically brittle base materials. While high compression rates yield considerable energy absorption capabilities, coiling provides a mechanical bi-stability and allows for reversible deformation. Consequently, the metamaterial has a promising potential for applications as a reversible energy absorber.

The bi-stability, however, relies on buckling of the longerons (see Fig. 5), which in general is known to be sensitive to material and geometrical imperfections. The complicated geometry of the unit cell can be effectively manufactured with 3D printing, where quality is a known issue since imperfections of the geometry cannot be completely eliminated. Alternatively, their effect can be mitigated during the design process. The referred article (Bessa et al. 2019) shows that this problem can be addressed with a data-driven approach using sparse GPs, which predict the uncertainty of the unit cell performance caused by imperfections. Here, we replicate this work and follow the same steps to learn the behavior of the unit cell with respect to energy absorption, but using sparse QGP algorithm instead of classical sparse GPs.

For demonstration purposes, however, we reduce the problem to only two out of the seven design parameters originally treated in the original article (Bessa et al. 2019) due to the high computational costs of classical simulation of quantum computation, which currently limits the QGP

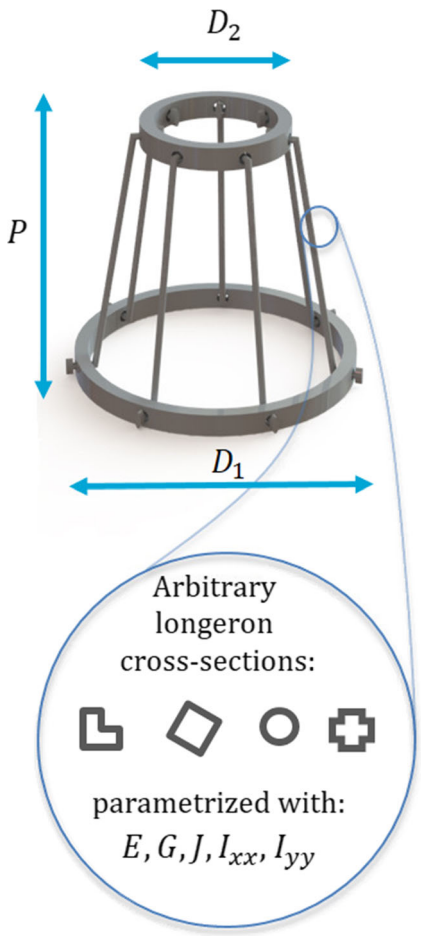

Fig. 5 A unit cell of the super-compressible metamaterial (Bessa et al. 2019)

systems to at most 8 training points (with reasonable accuracy) using standard computer hardware. The two selected running parameters are the cell's pitch (P) and the moment of inertia of the longeron around x axis (I_{xx}), both normalized by the diameter of the base ring D_1, following the convention from the original article (Bessa et al. 2019) (see also Fig. 5). The values of the remaining parameters were set based on the results provided in the original article, such that an interesting non-trivial and nonlinear demonstration case is obtained. Additional details are provided in the last section of the Supplementary Information. Note that we could have chosen any other example to illustrate SQGP.

Following the data-driven approach (Bessa et al. 2019), first the design space spanned by the two selected parameters was sampled according to the Sobol sequence. Specifically, we used 99 sample points which allowed to build a classical high-fidelity GP model acting as a ground-truth reference for the sparse QGP results. In the second step, the performance of each sample design is evaluated with a numerical analysis, specifically a finite element model simulating the unit cell under compression. This

yields the values of buckling load and energy absorption. The imperfections were simulated by introducing deviations in the input parameters of the finite element model, randomly drawn from a log-normal distribution (Bessa et al. 2019). The numerical analyses, however, indicated that certain architectures are non-coilable, but instead deform in a different manner. For the reversible energy absorption applications, however, only coilable designs are acceptable. Therefore, the design space was divided into coilable and non-coilable domains, using a classification model (see also the Supplementary Information). In the final step of the data-driven framework, the response of the material (i.e., the energy absorption and the critical buckling load) obtained with the computational analyses at the sample design points was used to train the machine learning algorithm. Based on the response at those few discrete locations, machine learning aims to predict the material response over the complete (continuous) design space. Such a model can be then used to find the point with optimal desired performance.

The coilable domain included 69 out of the total 99 sample points. Those 69 points were then used to generate a high-fidelity GP model used for reference. Next, a random set of 8 out of the 69 points was selected arbitrarily, based on its coverage of the coilable domain. Those 8 designs and their corresponding energy absorption and buckling load were used as training points for two models: the sparse QGP model, and the classical GP model used as a reference.

The predictions of the two models are compared in Fig. 6. The mean and uncertainty values are in good agreement,

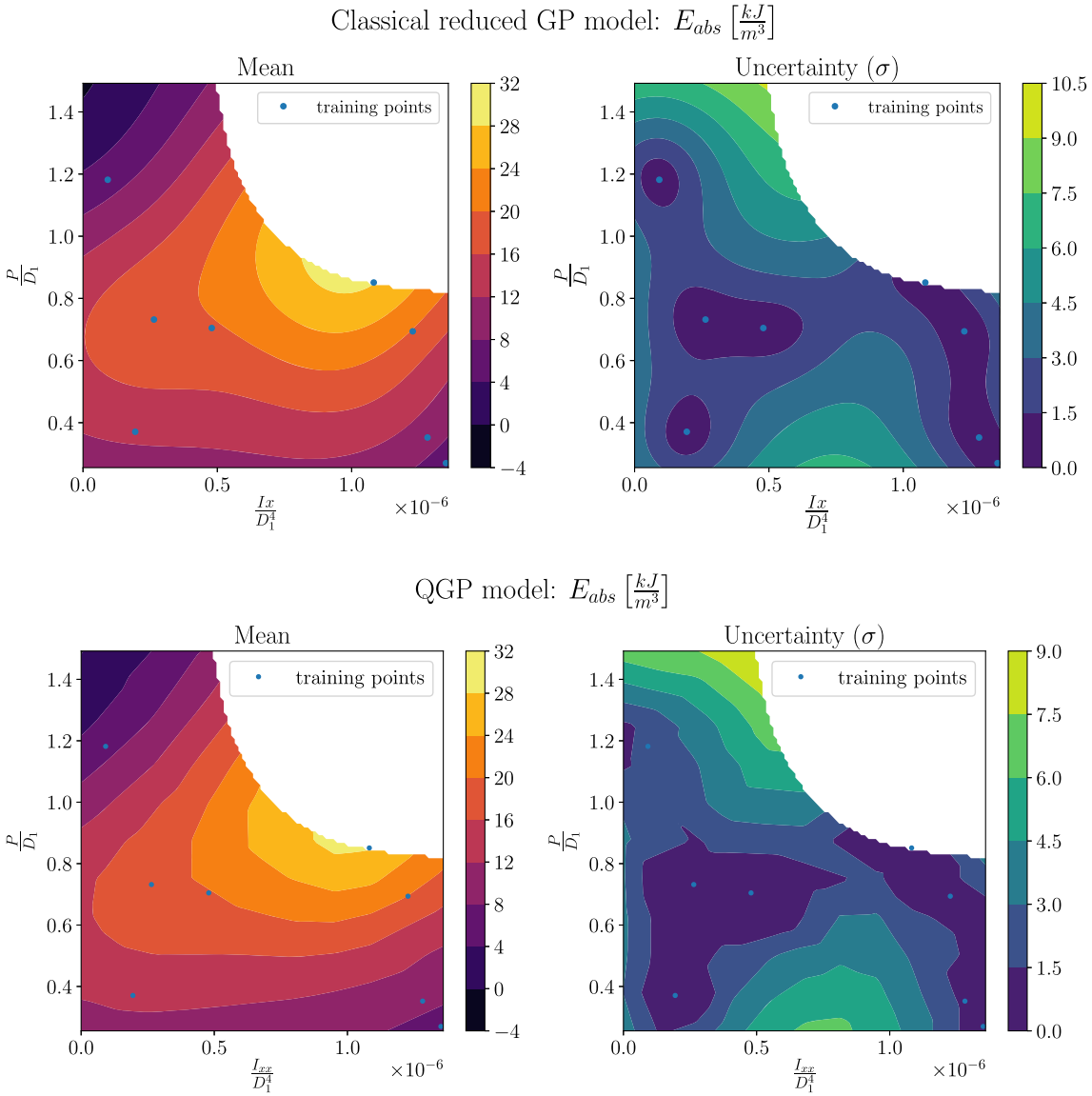

Fig. 6 Comparison of energy absorption for a unit cell predicted with quantum Gaussian processes algorithm with induced sparsity and classical GP model

but we discuss in the Supplementary Information additional details on the low-rank approximation obtained by QGP. The two models were also validated against the classical high-fidelity GP model, which was trained on the complete dataset of 69 points (only coilable domain is considered). Despite using only 8 points, the two models showed reasonable agreement. Additional comparisons with the models presented in the original article (Bessa et al. 2019) are included in the Supplementary Information. The presented examples prove that SQGP achieves comparable predictive performance, if not better, while achieving orders of magnitude improvement in scalability. This is of significant relevance for future applications of quantum machine learning.

5 Discussion and conclusion

In this work, we implemented QGP algorithm proposed by Zhao et al. (2019a) and explored its connection with the classical methods for improving scalability of GPs, specifically sparse GPs. We have demonstrated that quantum Gaussian processes have a natural mechanism for inducing sparsity, leading to unprecedented scalability when compared to classical full and sparse Gaussian processes, as well as other machine learning methods such as neural networks and their Bayesian versions. The results demonstrate that inducing sparsity in QGP algorithm allows reducing the computational cost by a few orders of magnitude, while preserving accuracy comparable to that achieved by the classical sparse GPs. The findings are supported by analytically derived error bounds and complexity analysis, as well as a numerical verification of the algorithm which revealed consistency with the analytical predictions.

From a broader perspective, our results provide new insights on quantum machine learning. We show that the common approach of replacing linear algebra routines with their quantum counterparts can additionally benefit from the features of quantum routines to gain a double advantage (accuracy and scalability). Furthermore, the implementation, and in particular the numerical cost model, of the QGP algorithm provides a contribution toward the costing problem of quantum algorithms, highlighted by Biamonte et al. (2017) as one of the key problems of contemporary quantum computing.

Our numerical implementation is limited by an inefficient Hamiltonian simulation which needs to be replaced in the future with a better scheme which allows reaching the exponential speed-up. One alternative for instance is the Hamiltonian simulation based on graph coloring method as proposed for the HHL algorithm (Childs 2010; Berry et al. 2007). The routine, however, requires strictly sparse matrices, which in general is not the case for GPs. This issue was already discussed elsewhere (Zhao et al. 2019a), who suggested addressing the problem with classical techniques by enforcing sparsity of the covariance matrix, e.g., by using compactly supported covariance functions (Buhmann 2001; Melkumyan and Ramos 2009). However, combining such functions with induced qPCA might lead to more intricate numerical effects, potentially with significant impact on the accuracy of the algorithm. Alternatively, non-sparse matrices could be tackled with techniques for dense matrices such as those proposed by Wossnig et al. (2018) and Das et al. (2018).

Despite those limitations our cost model provides good estimates of the overhead costs due to the supporting routines, which are commonly neglected in complexity analysis in literature. We show that those costs are significant even for relatively large systems. On the other hand, our results show that inducing sparsity in the QGP algorithm significantly reduces the overhead costs, which makes the algorithm more suitable for implementation on hardware with limited performance.

Finally, the presented proof-of-concept application example demonstrated that sparse quantum Gaussian processes can be successfully applied in data-driven design of materials, reaching levels of accuracy comparable to the current standards achieved with the classical sparse GPs. The scaling of SQGP allows matching the exponential growth of the design space, thus showing potential for overcoming the curse of dimensionality.

This research opens up a new path for application of quantum computing, which leverages quantum speed-ups to improve machine learning use in data-driven approaches. The concept has a promising potential for impact on materials science and beyond, yet the practical implementations remain limited by the technical performance of current quantum hardware.

Data availability The code and data used for the numerical tests presented in the article are available at: https://github.com/bessagroup/SQGP_SI.

Supplementary information The online version contains supplementary material available at (https://doi.org/10.1007/s42484-020-00032-8).

References

Acerbi L, Ji W (2017) Practical Bayesian optimization for model fitting with Bayesian adaptive direct search. In: Advances in neural information processing systems, pp 1836–1846

Ažman K, Kocijan J (2007) Application of Gaussian processes for black-box modelling of biosystems. ISA Transactions 46(4):443–457

Bartlett PL, Ben-David S (2002) Hardness results for neural network approximation problems. Theor Comput Sci 284(1):53–66

Belgacem E, Foffa S, Maggiore M, Yang T (2020) Gaussian processes reconstruction of modified gravitational wave propagation. Physical Review D 101(6):063505

Berry DW, Ahokas G, Cleve R, Sanders BC (2007) Efficient quantum algorithms for simulating sparse Hamiltonians. Commun Math Phys 270(2):359–371

Bessa M, Pellegrino S (2018) Design of ultra-thin shell structures in the stochastic post-buckling range using Bayesian machine learning and optimization. Int J Solids Struct 139:174–188

Bessa M, Bostanabad R, Liu Z, Hu A, Apley DW, Brinson C, Chen W, Liu WK (2017) A framework for data-driven analysis of materials under uncertainty: countering the curse of dimensionality. Comput Methods Appl Mech Eng 320:633–667

Bessa MA, Glowacki P, Houlder M (2019) Bayesian machine learning in metamaterial design: fragile becomes supercompressible. Advanced Materials 31(48):1904845

Biamonte J, Wittek P, Pancotti N, Rebentrost P, Wiebe N, Lloyd S (2017) Quantum machine learning. Nature 549(7671):195

Blum AL, Rivest RL (1992) Training a 3-node neural network is np-complete. Neural Netw 5(1):117–127

Buhmann M (2001) A new class of radial basis functions with compact support. Math Comput 70(233):307–318

Chalupka K, Williams CK, Murray I (2013) A framework for evaluating approximation methods for Gaussian process regression. J Mach Learn Res 14(Feb):333–350

Childs AM (2010) On the relationship between continuous-and discrete-time quantum walk. Commun Math Phys 294(2):581–603

Czekala I, Mandel KS, Andrews SM, Dittmann JA, Ghosh SK, Montet BT, Newton ER (2017) Disentangling time-series spectra with Gaussian processes: applications to radial velocity analysis. The Astrophysical Journal 840(1):49

Daniely A, Linial N, Shalev-Shwartz S (2014) From average case complexity to improper learning complexity. In: Proceedings of the forty-sixth annual ACM symposium on theory of computing, pp 441–448

Das S, Siopsis G, Weedbrook C (2018) Continuous-variable quantum Gaussian process regression and quantum singular value decomposition of nonsparse low-rank matrices. Physical Review A 97(2):022315

Frazier PI, Wang J (2016) Bayesian optimization for materials design. In: Information science for materials discovery and design. Springer, pp 45–75

Geiger M, Spigler S, d'Ascoli S, Sagun L, Baity-Jesi M, Biroli G, Wyart M (2019) Jamming transition as a paradigm to understand the loss landscape of deep neural networks. Physical Review E 100(1):012115

Geiger M, Jacot A, Spigler S, Gabriel F, Sagun L, d'Ascoli S, Biroli G, Hongler C, Wyart M (2020) Scaling description of generalization with number of parameters in deep learning. Journal of Statistical Mechanics: Theory and Experiment 2020(2):023401

Giovannetti V, Lloyd S, Maccone L (2008) Quantum random access memory. Phys Rev Lett 100(16):160501

Harrow AW, Hassidim A, Lloyd S (2009) Quantum algorithm for linear systems of equations. Phys Rev Lett 103:150502

Hatano N, Suzuki M (2005) Finding exponential product formulas of higher orders. In: Quantum annealing and other optimization methods. Springer, pp 37–68

Hensman J, Fusi N, Lawrence ND (2013) Gaussian processes for big data. arXiv:1309.6835

Hensman J, Matthews A, Ghahramani Z (2015) Scalable variational Gaussian process classification. J Mach Learn Res 38:351–360

Kumar S, Mohri M, Talwalkar A (2009) Sampling techniques for the nystrom method. In: Artificial intelligence and statistics, pp 304–311

Lee J, Xiao L, Schoenholz S, Bahri Y, Novak R, Sohl-Dickstein J, Pennington J (2019) Wide neural networks of any depth evolve as linear models under gradient descent. In: Advances in neural information processing systems, pp 8570–8581

Li L, Nayak N, Bian J, Baldi P (2020) Efficient neutrino oscillation parameter inference using Gaussian processes. Physical Review D 101(1):012001

Liu H, Ong YS, Shen X, Cai J (2018) When Gaussian process meets big data: a review of scalable GPs. arXiv:1807.01065

Livni R, Shalev-Shwartz S, Shamir O (2014) On the computational efficiency of training neural networks. In: Advances in neural information processing systems, pp 855–863

Lloyd S, Mohseni M, Rebentrost P (2014) Quantum principal component analysis. Nat Phys 10(9):631–633

Lookman T, Balachandran PV, Xue D, Hogden J, Theiler J (2017) Statistical inference and adaptive design for materials discovery. Curr Opinion Solid State Mater Sci 21(3):121–128

Melkumyan A, Ramos FT (2009) A sparse covariance function for exact Gaussian process inference in large datasets. In: Twenty-first international joint conference on artificial intelligence

Murphy KP (2012) Machine learning: a probabilistic perspective. MIT Press

Nielsen MA, Chuang IL (2000) Quantum computation and quantum information

Pearce T, Zaki M, Brintrup A, Anastassacos N, Neely A (2018) Uncertainty in neural networks: Bayesian ensembling. arXiv:1810.05546

Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E (2011) Scikit-learn: machine learning in python. J Mach Learn Res 12:2825–2830

Quiñonero-Candela J, Rasmussen CE (2005) A unifying view of sparse approximate Gaussian process regression. J Mach Learn Res 6(Dec):1939–1959

Seeger M (2004) Gaussian processes for machine learning. Int J Neural Syst 14(02):69–106

Shahriari B, Swersky K, Wang Z, Adams RP, De Freitas N (2015) Taking the human out of the loop: a review of Bayesian optimization. Proc IEEE 104(1):148–175

Snelson E, Ghahramani Z (2006) Sparse Gaussian processes using pseudo-inputs. In: Advances in neural information processing systems, pp 1257–1264

Tancret F, HKDH B, DJC M (1999) Comparison of artificial neural networks with gaussian processes to model the yield strength of nickel-base superalloys. ISIJ International 39(10):1020–1026

Titsias M (2009) Variational learning of inducing variables in sparse Gaussian processes. In: Artificial intelligence and statistics, pp 567–574

van der Walt S, Colbert SC, Varoquaux G (2011) The numpy array: a structure for efficient numerical computation. Computing in Science & Engineering 13(2):22–30

Wang S, Zhang Z (2014) Efficient algorithms and error analysis for the modified Nystrom method. In: Artificial intelligence and statistics, pp 996–1004

Wathen AJ, Zhu S (2015) On spectral distribution of kernel matrices related to radial basis functions. Numerical Algorithms 70(4):709–726

Williams CK, Rasmussen CE (2006) Gaussian processes for machine learning, vol 2. MIT Press, Cambridge

Williams CK, Seeger M (2001) Using the Nyström method to speed up kernel machines. In: Advances in neural information processing systems, pp 682–688

Wilson A, Nickisch H (2015) Kernel interpolation for scalable structured gaussian processes (kiss-gp). In: International conference on machine learning, pp 1775–1784

Wilson AG, Izmailov P (2020) Bayesian deep learning and a probabilistic perspective of generalization. arXiv:2002.08791

Wossnig L, Zhao Z, Prakash A (2018) Quantum linear system algorithm for dense matrices. Phys Rev Lett 120(5):050502

Zhao Z, Fitzsimons JK, Rebentrost P, Dunjko V, Fitzsimons JF (2018) Smooth input preparation for quantum and quantum-inspired machine learning. arXiv:1804.00281

Zhao Z, Fitzsimons JK, Fitzsimons JF (2019a) Quantum-assisted gaussian process regression. Physical Review A 99(5):052331

Zhao Z, Pozas-Kerstjens A, Rebentrost P, Wittek P (2019b) Bayesian deep learning on a quantum computer. Quantum Machine Intelligence 1(1-2):41–51